Libra : une méthode de médiation auto-adaptative en fonction des attentes des participants

Philippe Lamarre, Jorge-Arnulfo Quiane-Ruiz, Patrick Valduriez

Libra : une méthode de médiation auto-adaptative en fonction des attentes des participants

Philippe Lamarre
lamarre@univ-nantes.fr

Jorge-Arnulfo Quiané-Ruiz
quiane@univ-nantes.fr

Patrick Valduriez
Patrick.Valduriez@inria.fr

Atlas group, INRIA & LINA
Université de Nantes
2 rue de la Houssinière, 44322 Nantes – FRANCE

Résumé

Nous considérons le problème de l'allocation de tâches dans le cadre d'environnements ouverts où les participants (fournisseurs et clients) ont des attentes particulières. De nombreux travaux sur l'allocation de tâches se sont concentrés sur des problèmes de maximisation ou de minimisation de fonctions prédéfinies (temps de réponse, répartition de charge...). Cependant, ces objectifs sont définis de manière globale alors que les participants peuvent avoir des objectifs différents, voire divergents. Nous proposons ici une méthode de médiation, $Libra$, qui s'auto-adapte aux attentes des participants. $Libra$ effectue les allocations en fonction des intérêts individuels des fournisseurs et des clients. Cette prise en compte est régulée par les satisfactions individuelles de sorte à obtenir une certaine équité. Des simulations nous permettent de comparer $Libra$ avec d'autres techniques existantes (répartition de charge, allocation de tâche dans le style de Mariposa). Les résultats montrent que, tout en conservant un haut degré d'efficacité (e.g. temps de réponse), $Libra$ permet aux participants d'atteindre un degré de satisfaction bien supérieur.

Mots-clés : *Allocation de tâches, participants autonomes, satisfaction des participants*

Abstract

We consider the task allocation problem in open environments where providers have special interests for some tasks and consumers have some expectations w.r.t. the results they get. We propose $Libra$, a mediation mechanism that adapts online to the consumer and providers' expectations. We implemented $Libra$ and compared it to two important baseline methods for allocating tasks, namely $Capacity\ based$ and Mariposa-like. The results show that $Libra$ significantly outperforms the baseline methods by ensuring high efficiency and satisfying participants.

Keywords: *Task allocation, autonomous participants, participants' satisfaction*

1 Introduction

Les systèmes d'information distribués font souvent l'hypothèse que leurs participants[1] sont autonomes, c'est-à-dire qu'ils sont libres de rejoindre ou de quitter le "système"[2] à n'importe quel moment et sans avoir à en référer à qui que ce soit. La motivation des participants à intégrer le système peut être liée à l'espoir que ce système peut répondre à leurs attentes et leur permettre d'atteindre leurs objectifs. Le départ quand à lui est souvent consécutif à la déception. Pour le bon fonctionnement de ces systèmes, il est donc primordial que la répartition des tâches soit attentive aux attentes des participants, tant des clients que des fournisseurs, de sorte à ce qu'ils soient *satisfaits*, autant que faire ce peut.

De nombreux travaux ont été menés dans le contexte de l'allocation de tâches : de la recherche des fournisseurs pouvant réaliser une tâche [7, 10], à l'allocation d'une tâche de sorte à maximiser ou minimiser certains critères comme la répartition de charge et le temps de réponse [2, 5, 9, 13, 16]. Cependant, les attentes des participants se joignant au système ne sont pas nécessairement restreintes aux performances. Les clients peuvent manifester un certain intérêt concernant la qualité des résultats (si tous les fournisseurs ne donnent pas les mêmes réponses). Parmi les tâches qu'ils peuvent traiter, les fournisseurs peuvent avoir des préférences pour certaines. Tous peuvent avoir des préférences concernant les agents avec lesquels ils traitent. L'*intention* qu'a un fournisseur de traiter une tâche est donc le résultat de la combinaison de plusieurs critères qui lui sont propres.

De même, l'*intention* d'un client à voir sa tâche traitée par tel ou tel fournisseur est aussi le résultat de considérations personnelles. Cette ap-

[1] Tout au long de cet article, le terme participant fait référence à la fois aux fournisseurs et aux clients.

[2] Le "système" peut désigner soit le système d'information distribué, soit plus localement un médiateur [1, 14].

proche soulève un problème. Si aucun fournisseur ne manifeste d'intérêt pour une tâche donnée, elle risque de ne pas être traitée. C'est de fait ce qui se produit dans différents systèmes basés sur des techniques de micro-économie [3, 4, 17]. Dans un tel cas, le traitement de la tâche doit être imposé à un ou plusieurs fournisseurs, ce qui les mécontentera (sauf à les dédommager). D'un autre coté, si la tâche n'est pas traitée, c'est le client qui sera mécontent.

Les problèmes inhérents à l'allocation de tâches sont donc de nature differente. D'abord, les attentes des participants peuvent-être contradictoires. Ensuite, le fait que nous considérons que les tâches doivent être traitées par le système, même si les fournisseurs ne souhaitent pas les traiter pour des raisons qui leurs sont propres, peut introduire du mécontentement. Enfin, trop de mécontentement conduit les participants à quitter le système ce qui peut avoir des conséquences sur les fonctionnalités offertes par le système. Le départ de fournisseurs peut conduire à perdre des fonctionnalités, et le départ de clients est une perte de source de travail pour les fournisseurs.

A notre connaissance, ce problème n'a pas été adressé dans son ensemble. Les mécanismes de médiation qui effectuent l'allocation de tâches ne tiennent compte ni des *intentions* des participants, ni de leur *satisfaction*. Les contributions majeures de cet article sont donc :

– La proposition d'un mécanisme de médiation ($Libra$) qui s'adapte immédiatement et automatiquement aux attentes des participants. Ce mécanisme utilise les *intentions* exprimées par les participants pour définir leur *satisfaction*. Il utilise ensuite ces deux notions pour allouer les tâches. Pour atteindre une certaine équité, $Libra$ pondère les intentions des différents participants en fonction de leurs satisfactions respectives.
– Une analyse des techniques d'allocation du point de vue de la satisfaction.
– Une validation expérimentale comparant $Libra$ à d'autres techniques existantes ($Capacity\ based$ et *Mariposa-like*) qui montre la supériorité de notre approche.

La suite de cet article est organisée comme suit. La section 2 présente des travaux ayant déjà traité cette problématique. Les concepts de base sont énoncés dans la section 3. La section 4 est consacrée à la présentation de notre approche. Des résultats expérimentaux ainsi que de comparaisons aux approches existantes sont présentés dans la section 5, avant de conclure dans la section 6.

2 Travaux précédents

Dans le contexte des systèmes d'information distribués à grande échelle, de nombreuses approches se sont concentrées sur le problème de l'allocation de tâches avec comme objectif les performances du système, sans aucune considération des intentions des participants, sauf à considérer qu'ils sont eux-mêmes exclusivement intéressés par les performances. Par exemple, les propositions [13, 16] allouent chaque tâche entrante aux fournisseurs qui sont les moins chargés parmi ceux qui peuvent traiter la tâche.

Les techniques de médiation utilisant une approche économique peuvent prétendre prendre en compte les intentions. Mariposa [17] est l'un des premiers systèmes utilisant des techniques de microéconomie pour la gestion des informations dans un système réparti. Cette proposition est basée sur un système de vente aux enchères. Pour schématiser, les clients payent les fournisseurs pour qu'ils traitent leurs tâches. Le prix établi par les fournisseurs est calculé en fonction de leurs préférences et de leur charge de travail, ceci afin de garantir un certain équilibre de la charge de travail au sein du système. Une fois les offres publiées, un broker sélectionne les fournisseurs ayant les offres les plus basses. En revanche, nos expérimentations montrent que Mariposa ne garantit pas un bon équilibre de charge de tâches. En outre, si aucun fournisseur ne souhaite traiter la tâche, elle ne le sera tout simplement pas.

La médiation, dite *médiation flexible*, proposée dans [6] est aussi basée sur des aspects économiques. Là encore, les fournisseurs font des offres pour obtenir des tâches. Ces offres sont alors équilibrées par leur qualité estimée (ou réputation). Le prix que doit payer un fournisseur pour obtenir la tâche dépend de sa réputation. Contrairement à Mariposa, lorsqu'une tâche n'intéresse personne, cela conduit à la "réquisition" de fournisseurs auxquels on impose de la traiter. Un mécanisme de compensation financière est alors mis en œuvre. Cette compensation augmente les possibilités de ces fournisseurs à faire valoir leurs choix dans les prochains tours. Dans cette approche, ce sont des mécanismes économiques qui sont utilisés pour réguler le système, la satisfaction des participants étant supposée être une propriété induite.

Dans [11], nous avons proposé une technique de médiation basée sur la satisfaction des four-

nisseurs, mais ni la satisfaction des clients, ni leur intentions ne sont prise en compte. Dans [12], nous avons proposé une stratégie pour prendre en compte ces différentes notions, mais aucune méthode pour faire la fusion des *intentions* des clients et fournisseurs n'est proposée. En outre, les stratégies que nous proposons dans [12] peuvent être utilisées pour améliorer les résultats de la proposition de cet article.

3 Notions préliminaires

Le système que nous considérons est constitué d'un ensemble de fournisseurs P, d'un ensemble de clients C et d'un ensemble de médiateurs M. Ces ensembles ne sont pas nécessairement disjoints, un même agent pouvant jouer plusieurs rôles. Les fournisseurs peuvent être hétérogènes en termes de capacités, ne disposant pas tous des mêmes ressources, mais aussi en termes de données. Ce dernier point signifie qu'ils peuvent donner des résultats différents les uns des autres pour la même tâche. Les tâches sont abstraites par un triplet $q = < c, d, n >$ tel que $q.c \in C$ est l'identifiant du client ayant émis la tâche, $q.d$ est la description de la tâche, et $q.n \in \mathbb{N}^*$. Le paramètre $q.n$ représente le nombre de fournisseurs que le client veut voir traiter la tâche. Considérons par exemple une application de commerce électronique. Dans ce cas, la tâche correspond à un appel à propositions. Donc, lorsqu'un client fait un appel à propositions, il peut souhaiter limiter le nombre de réponses à n. De même, les fournisseurs ne souhaitent généralement pas répondre à tous les appels d'offres.

La tâche peut être exprimée de manière textuelle, logique, dans un langage spécifique tel que SQL, XQuery, *etc.* Le problème de "matchmaking", consistant à identifier les fournisseurs pouvant travailler avec cette requête est en dehors du champ de cet article (voir [7, 10] pour plus d'informations). Nous nous contentons de supposer que nous disposons d'un processus permettant d'identifier les fournisseurs adéquats de manière idéale, i.e. sans faux positif ni faux négatif.

Les clients confient leurs tâches à un médiateur $m \in M$ dont le rôle est d'allouer chaque tâche q à $q.n$ fournisseurs. P_q représente l'ensemble des fournisseurs associés au médiateur m (n'apparaissant pas dans la notation pour éviter de l'alourdir) pouvant traiter la tâche q. Cet ensemble de fournisseurs est obtenu via un processus de "matchmaking" supposé correct.

L'allocation d'une tâche q est formalisée par un vecteur $All\overrightarrow{oc}_q$, ou $All\overrightarrow{oc}$ s'il n'y a pas d'ambiguïté sur q, de longueur N tel que :

$$\forall p \in P_q,\ All\overrightarrow{oc}[p] = \begin{cases} 1 & \text{si la tache est allouée à } p \\ 0 & \text{sinon} \end{cases}$$

Dans le cas où le nombre de fournisseurs pouvant traiter la tâche est insuffisant par rapport au nombre de fournisseurs demandés par le client, ils doivent tous la traiter. Ceci impose donc que $\sum_{p \in P_q} All\overrightarrow{oc}[p] = min(q.n,\ N)$ où $N = ||P_q||$.

3.1 Intentions des participants

Les *intentions* des participants sont exprimées sur l'intervalle $[-1..1]$. Une intention positive traduit le souhait que le choix devienne réalité, souhait d'autant plus important que la valeur est proche de 1. Au contraire, une intention négative traduit le souhait de ne pas voir cette possibilité se réaliser, souhait d'autant plus important que la valeur est proche de -1. Une valeur de 0 traduit une indifférence.

Il est de la responsabilité d'un participant de calculer ses propres intentions en combinant les critères qu'il juge utile de considérer (e.g. charge, préférences, temps de réponse, réputation, expériences passées,...). La manière dont les participants calculent leurs *intentions* est considérée comme une information privée à laquelle le système ne peut accéder. Cependant, il ne faut pas s'y tromper, cela a un impact direct sur le comportement global du système. Par exemple, si les participants manifestent tous un intérêt marqué pour des temps de réponse les plus faibles possibles, la médiation tenant compte de leurs intentions devrait conduire à l'obtention d'un système performant du point de vue des temps de réponse. Tel ne sera pas le cas si les participants s'intéressent à la qualité des réponses sans aucune considération pour le temps. La médiation doit permettre d'adapter le comportement global du système aux attentes des participants.

3.2 Satisfaction d'un participant

Intuitivement, la *satisfaction* d'un client est liée aux fournisseurs qui ont traité ses tâches : si ces fournisseurs correspondent à ses attentes il doit être satisfait ; sinon, il ne l'est pas. Similairement, la *satisfaction* d'un fournisseur est relative aux tâches qui lui ont été allouées. Il doit être d'autant plus satisfait qu'elles correspondent à ses attentes.

La *satisfaction* s'exprime donc en fonction des événements passés. Nous supposons avoir des agents avec des mémoires limitées, nous considérons donc qu'ils ne mémorisent que les k derniers événements (pour des raisons de simplification, k est supposé identique pour tous les participants).

Intuitivement, les participants peuvent évaluer eux-mêmes leur satisfaction. Ils disposent pour cela de toutes les informations nécessaires. Le système de médiation pourrait donc se contenter de leur demander de bien vouloir fournir cette information. Cependant, ce sont des données particulièrement sensibles (très personnelles) dont l'impact sur les médiations à venir est important. Les raisons de fournir de fausses informations sont donc nombreuses et d'importance. Le système de médiation a donc peu de chance d'obtenir des informations fiables par cette méthode. Si le mécanisme de médiation dispose des *intentions* affichées par les participants, il peut les utiliser pour évaluer le degré de satisfaction des participants (en supposant que leurs intentions sont relativement accordées avec les résultats effectivement obtenus). C'est donc à partir des *intentions* exprimées que les *satisfactions* sont calculées par le médiateur.

Satisfaction des clients. Avant de pouvoir nous intéresser à la *satisfaction* d'un client concernant les k dernières allocations, il est nécessaire de savoir calculer la *satisfaction* relative à l'allocation d'une tâche donnée. La moyenne des *intentions* exprimées par le client concernant les fournisseurs affectés au traitement de la tâche q semble être une candidate intuitivement naturelle. C'est cette solution que nous avons retenue à un détail près. Pour tenir compte du nombre de fournisseurs souhaités par le client, la somme des intentions exprimées concernant les fournisseurs choisis est divisée non pas par le nombre de fournisseurs choisis, mais par le nombre de fournisseurs désirés par le client. Ainsi la satisfaction est d'autant plus faible que le nombre de fournisseurs affectés au traitement de la tâche par la méthode d'allocation sera faible par rapport au nombre demandé par le client. La satisfaction d'un client c pour sa tâche q est donc :

$$\delta_s(c,q) = \Big((\frac{1}{n} \sum_{p \in \widehat{P_q}} \overrightarrow{CI}_c^q[p]\ ) + 1 \Big) \Big/ 2 \qquad (1)$$

où n est une notation abrégée de $q.n$ et $\overrightarrow{CI}_c^q[p]$ contient les intentions du client c concernant l'allocation de sa tâche pour chaque fournisseur p de $\widehat{P_q}$. Pour avoir des notations plus synthétiques, nous introduisons $\widehat{P_q}$ qui représente l'ensemble des fournisseurs auxquels la tâche a été affectée. Notons que, par construction, les valeurs de $\delta_s(c,q)$ sont dans l'intervalle $[0..1]$.

Soit, IQ_c^k les k dernières tâches ayant été soumises par le client c. Selon notre hypothèse concernant la mémoire limitée d'un agent, cela correspond à l'ensemble des tâches qu'il est capable de mémoriser. La *satisfaction* d'un client c concernant les allocations réalisées sur ses k dernières tâches est simplement la moyenne de sa satisfaction concernant l'allocation de chacune de ces tâches.

Définition 1 *Satisfaction d'un client*

$$\delta_s(c) = \frac{1}{k} \sum_{q \in IQ_c^k} \delta_s(c,q)$$

Notons là encore que par construction, la valeur de $\delta_s(c)$ est dans l'intervalle $[0..1]$. Plus la valeur est proche de 1, plus l'agent est satisfait.

Satisfaction des fournisseurs. Dans un premier temps, remarquons que si le client n'est concerné que par les tâches qu'il émet, le fournisseur quant à lui est concerné par toutes les tâches qu'il peut traiter (qu'il soit sélectionné ou non pour les traiter). Soit donc PQ_p^k l'ensemble des k dernières tâches qui ont été "proposées" au fournisseur p. D'après l'hypothèse concernant la mémoire limitée d'un agent, cela correspond à l'ensemble des tâches que l'agent p peut mémoriser. Parmi ces tâches, l'ensemble de celles que l'agent p a à traiter (celles que le processus de médiation lui affecte) est noté $SQ_p^k \subseteq PQ_p^k$. La *satisfaction* de p est alors obtenue en faisant la moyenne des *intentions* qu'il a montrées concernant les tâches qu'il a effectivement eues à traiter, i.e.

Définition 2 *Satisfaction du fournisseur*

$$\delta_s(p) = \begin{cases} \Big((\frac{1}{||SQ_p^k||} \sum_{q \in SQ_p^k} \overrightarrow{PI}_p[q]\ ) + 1 \Big) \Big/ 2 & \\ 0 & si\ SQ_p^k = \emptyset \end{cases}$$

avec $\overrightarrow{PI}_p[q]$ représentant l'*intention* de p pour traiter la tâche q. Notons que par construction, les valeurs de $\delta_s(p)$ sont dans l'intervalle $[0..1]$.

4 Mécanisme de médiation

$Libra$ est un mécanisme d'allocation fondé sur la prise en compte des *intentions* et de la *satisfaction* des participants. Cela lui permet de s'adapter immédiatement et automatiquement aux changements d'*intentions* des participants. Par exemple, le système de médiation ne tiendra compte des performances (temps de réponse et répartition de charge) que si les participants en tiennent eux mêmes compte dans l'expression de leurs intentions.

Nous présentons ici $Libra$ en deux phases. La première partie décrit la technique d'évaluation d'un score pour chaque fournisseur correspondant à la pertinence de lui allouer la tâche. La deuxième partie présente l'algorithme général de $Libra$.

4.1 Pertinence d'allouer une tâche à un fournisseur

Étant donné une tâche q et un fournisseur p, la pertinence d'allouer cette tâche à ce fournisseur est évaluée et quantifiée en considérant les deux points de vue en présence. Le point de vue du client c est obtenu en considérant son *intention* de voir sa tâche traitée par p et le point de vue du fournisseur p est obtenu en considérant son *intention* de traiter la tâche q de c. La confrontation de ces deux points de vue pourrait être directe, mais dans un souci d'équité, nous avons choisi de permettre qu'un point de vue soit privilégié par rapport à l'autre.

Définition 3 *Évaluation d'un fournisseur*

$$Scr_q(p) = \begin{cases} (\overrightarrow{PI}_q[p])^{\omega} (\overrightarrow{CI}^q_c[p])^{1-\omega} \\ \qquad si\ \overrightarrow{PI}_q[p] > 0 \wedge \overrightarrow{CI}^q_c[p] > 0 \\ -\Big(((1 - \overrightarrow{PI}_q[p]) + \epsilon)^{\omega} ((1 - \overrightarrow{CI}^q_c[p]) + \epsilon)^{1-\omega}\Big) \\ \qquad sinon \end{cases}$$

À différence du vecteur $\overrightarrow{PI}_p$ qui stocke les *intentions* du fournisseur p pour taiter les dernières tâches qui lui ont été proposées, le vecteur $\overrightarrow{PI}_q$ stocke l'*intention* de chaque fournisseur $p \in P_q$ pour traiter la tâche q (mêmes informations notées différemment suivant les points de vue adoptés : fournisseur ou mediateur). Le paramètre $\epsilon > 0$ est une constante habituellement valuée à 1. Son rôle est seulement d'éviter le passage à zéro. ω est une variable qui prend ses valeurs dans l'intervalle $[0..1]$; elle traduit le fait qu'un point de vue peut

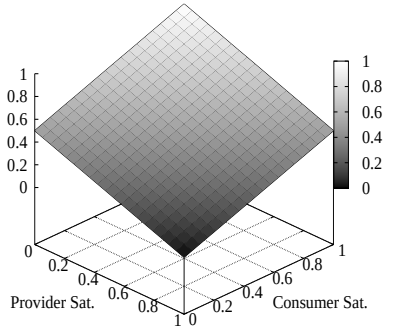


FIG. 1 – Valeurs possibles de ω en fonction des satisfactions.

être privilégié par rapport à l'autre. Lorsque ω vaut $0,5$, les deux points de vue sont considérés avec une égalité parfaite. Lorsque ω vaut 0, le point de vue du fournisseur est totalement occulté pour ne prendre en compte que celui du client. L'inverse est vrai quand ω vaut 1.

Pour chaque évaluation $Libra$ calcule la valeur de ω en fonction de la *satisfaction* des agents dont on confronte le point de vue. L'idée est de privilégier le point de vue de l'agent qui est le moins satisfait. ω représente donc la différence de satisfaction entre le fournisseur et le client en présence. La Figure 1 illustre les valeurs que ω peut prendre selon la *satisfaction* des clients et fournisseurs.

$$\omega = \Big(\big(\delta_s(c) - \delta_s(p)\big) + 1\Big) \Big/ 2 \qquad (2)$$

Une fois quantifiées la pertinence d'allouer la tâche pour chaque fournisseur pouvant traiter la tâche, il n'est pas difficile d'ordonner les fournisseurs du plus pertinent au moins pertinent. Le résultat de cet ordonnancement est mémorisé dans un vecteur $\overrightarrow{R}^q$, où $\overrightarrow{R}^q[1]$ (respectivement $\overrightarrow{R}^q[N]$) est le fournisseur le plus (respectivement le moins) pertinent.

4.2 Allocation de tâches

Pour allouer une tâche q, dans un premier temps, le médiateur doit être capable de déterminer l'ensemble de fournisseurs qui ont la capacité de traiter cette tâche (i.e. l'ensemble P_q). Un grand nombre de travaux ont déjà porté sur ce problème, voir par exemple [7, 10]. Aussi nous considérerons ce problème comme étant résolu par des techniques que nous ne présentons pas ici.

Les grandes étapes permettant d'allouer la tâche q à $q.n$ fournisseurs parmi ceux de l'ensemble P_q sont présentées dans l'algorithme 1 .

Algorithm 1: Allocation d'une tâche

Input : q, P_q
Output: $All\overrightarrow{oc}_q$

1 **begin**
  `// Intention du client`
2 **fork** demander l'intention à $q.c$;
  `// Ints. des fournisseurs`
3 **foreach** $p \in P_q$ **do**
4 **fork** demander l'intention à p ;
5 **waituntil** $\overrightarrow{CI}_c^q$ et $\overrightarrow{PI}_q$ ou un *timeout*;
  `// Éval. des fournisseurs`
6 **foreach** $p \in P_q$ **do**
7 évaluer p par rapport $\overrightarrow{CI}_c^q$ & $\overrightarrow{PI}_q$;
  `// Ordre des fournisseurs`
8 rank P_q, $\overrightarrow{R}^q$, par rapport $Scr_p(q)$;
  `// Selection des fournisseurs`
9 **for** $i = 1$ **to** $min(n, N)$ **do**
  $All\overrightarrow{oc}[\overrightarrow{R}^q[i]] \leftarrow 1$;
10 **for** $j = min(n, N) + 1$ **to** N **do**
  $All\overrightarrow{oc}[\overrightarrow{R}^q[j]] \leftarrow 0$;
11 **end**

Après avoir demandé en parallèle et obtenu les *intentions* des différents participants concernés par cette allocation (ligne 2)[3], l'évaluation de chaque fournisseur est calculée (ligne 7) comme indiqué dans la section précédente. Une fois triés (ligne 8), ceux qui sont considérés comme étant les plus pertinents sont sélectionnés (ligne 9). Finalement, tous les participants à cette médiation sont informés du résultat (lignes 9 et 10). Cet algorithme peut être optimisé de bien des manières, mais notre but est ici d'en présenter une version facilement compréhensible.

5 Validation

L'objectif principal est d'analyser comment les méthodes d'allocation tiennent compte des notions de satisfaction (avec un intérêt particulier pour $Libra$). Pour cela, nous avons procédé en deux temps. Dans une première étape, nous avons mesuré les *satisfactions* en considérant que les participants sont captifs (aucune possibilité de quitter le système). Puis dans un second temps, nous avons donné la possibilité aux participants de quitter le système pour étudier l'impact de cette autonomie.

[3]Un timeout évite les attentes trop longues.

5.1 Paramètres des simulations

En utilisant SimJava, nous avons développé en Java un simulateur pour représenter un système d'information distribué comme défini dans [6]. Pour toutes les méthodes que nous avons testées, la configuration est la même (c.f. Tableau 1). Seule la technique d'allocation diffère.

Nous initialisons les participants avec une valeur de satisfaction de 0.5 qui évolue avec les 200 tâches entrantes et les 500 tâches proposées. Autrement dit, la valeur du paramètre k est 200 pour les clients et 500 pour les fournisseurs. Le nombre de clients, fournisseurs et médiateurs dans le système est 200, 400 et 1, respectivement. Nous avons affecté les ressources suffisantes au médiateur de sorte qu'il ne cause pas de goulot d'étranglement dans le système. L'*utilisation* d'un fournisseur p à un moment donné t (notée par la fonction $U_t(p)$) dénote la charge de p à t. Inspirés de [5], nous supposons que les fournisseurs obtiennent leur *utilisation* comme l'équation 3 où la fonction $cost_p(q)$ dénote le coût de traitement de la tâche q par le fournisseur p et la fonction $cap(p)$ dénote la capacité de traitement du p.

$$U_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap(p)} \quad (3)$$

D'une part, nous supposons qu'un fournisseur calcule son *intention* pour traiter une tâche q comme dans [11] (voir équation 4). Cette technique permet à un fournisseur de prendre en compte à la fois son *utilisation* et ses *préférences* en portant une attention plus ou moins soutenue à l'une ou l'autre en fonction de sa *satisfaction* actuelle. Pour cela, à la différence de la section 3.2, la satisfaction $\delta_s(p)$ utilisée ici est basée sur les *préférences*, notées $Prf_p(q)$).

$$PI_p(q) = \begin{cases} (Prf_p(q)^{1-\delta_s(p)})(1-U_t(p))^{\delta_s(p)}, \\ \qquad si (Prf_p(q) > 0) \wedge (U_t(p) < 1) \\ -\Big(\big((1-Prf_p(q))+\epsilon\big)^{1-\delta_s(p)}\big(U_t(p)+\epsilon\big)^{\delta_s(p)}\Big) \\ \qquad sinon \end{cases} \quad (4)$$

D'autre part, pour simuler une hétérogénéité élevée des *intentions* chez les clients, nous divisons l'ensemble de fournisseurs en trois classes selon l'intérêt des clients : ceux pour qui les clients ont un grand intérêt (60% des fournisseurs), un intérêt moyen (30% des fournisseurs) et un intérêt faible (10% des fournisseurs). Par simplicité, nous nommons ces groupes de fournisseurs les *très-intéressants*,

TAB. 1 – Paramètres des simulations.

Paramètre	Définition	Valeur
nbConsumers	Nombre de clients	200
nbProviders	Nombre de fournisseurs	400
nbMediators	Nombre de médiateurs	1
qDistribution	Distribution dans laquelle les tâches arrivent au sytème	Poisson
iniSatisfaction	Satisfaction initiale	0.5
conSatSize	Valeur de k pour les clients	200
proSatSize	Valeur de k pour les fournisseurs	500
nbRepeat	Nombre de répétitions par simulation	10

moyennement-intéressants et *peu-intéressants* respectivement. Les clients obtiennent leurs *intentions* envers les fournisseurs *très-intéressants* entre $0,34$ et 1, envers les fournisseurs *moyennement-intéressants* entre $-0,54$ et $0,34$ et envers les fournisseurs *peu-intéressants* entre -1 et $-0,54$. Sans perte de généralité, nous pourrions utiliser d'autres mécanismes pour obtenir les *intentions* des clients (par exemple, en utilisant les langages *TCL* ou *Rush*). Nous nous basons sur les résultats présentés dans [15] pour paramétrer l'hétérogénéité des capacités des fournisseurs. 10% des fournisseurs ont une capacité faible, 60% des fournisseurs ont une capacité moyenne et 30% des fournisseurs ont une capacité forte. Par simplicité, nous nommons ces trois groupes de fournisseurs les *très-capables*, *moyennement-capables* et *peu-capables*. Les fournisseurs *très-capables* sont trois fois plus puissants que les fournisseurs *moyennement-capables* et sept fois plus que les fournisseurs *peu-capables*. Finalement, nous produisons deux classes de tâches qui sont traitées par les fournisseurs *très-capables* dans un temps de 1.3 et 1.5 secondes, respectivement. Les tâches arrivent au système avec une distribution de Poisson, couramment utilisée dans des environnements ouverts [8].

Dans cet article, nous ne considérons pas le problème de la bande passante et nous supposons que tous les fournisseurs disposent des mêmes capacités réseau. Finalement, nous supposons que les clients ne demandent qu'un seul résultat par tâche et que tous les fournisseurs dans le système peuvent satisfaire toute tâche entrante (pas de problème de "matchmaking").

5.2 Méthodes de référence

Capacity based. Dans le contexte des systèmes d'information distribués, les deux techniques les plus connues pour faire de l'allocation de tâches sont basées sur la charge [2, 5] et la capacité [9, 13, 16]. Nous ne considérons pas les méthodes basées sur la charge car, à la différence de celles basées sur la capacité, elles supposent que tous les fournisseurs et toutes les tâches sont homogènes. Le principe des méthodes basées sur la capacité est d'affecter chaque tâche entrante aux fournisseurs qui sont les moins utilisés parmi ceux de l'ensemble P_q. $Capacity\ based$ s'est avéré meilleur que $Load\ based$ dans des systèmes hétérogènes. Donc, nous comparons $Libra$ à cette approche (que nous nommons $Capacity\ based$ pour simplicité) dans nos simulations.

Mariposa. Diverses approches économiques ont été proposées [3, 4, 17] pour faire de l'allocation de tâches. Comme présenté dans la section 2, Mariposa [17] est l'un des premiers systèmes utilisant des techniques de micro-économie. Mariposa a montré de bonnes performances dans des environnements ouverts et hétérogènes. C'est pourquoi nous l'avons implémenté et comparé à notre proposition. Dans Mariposa, chaque tâche entrante q arrive à un médiateur (broker) qui trouve l'ensemble P_q et demande à chaque fournisseur de P_q son *offre* pour traiter q. Les fournisseurs calculent leurs offres en fonction de leurs préférences et de leur charge actuelle. Une fois ces *offres* obtenues le médiateur alloue la tâche aux fournisseurs ayant fait l'*offre* la plus basse.

5.3 Résultats expérimentaux

Si les participants sont autonomes, ils peuvent quitter le système par *mécontentement* ou *famine*. Néanmoins, le choix du seuil de départ est très subjectif et peut dépendre de nombreux facteurs. Nous supposons que les participants dans le système supportent des seuils élevés de *mécontentement* et de *famine*. Un client décide de quitter le système par *mécontentement* si sa *satisfaction* est inférieure à 0.5, c'est-à-dire si les allocations ne lui sont pas favorables. D'autre part, un fournisseur décide de quitter le système par *mécontentement* si sa *satisfaction* est inférieure à 0.35, respectivement par *famine* si son *utilisation* est inférieure de 20% à son *utilisation* théorique. L'utilisation théorique d'un fournisseur est égale à la charge totale du système (i.e. si le système présente une charge de 80%, chaque fournisseur devrait être chargé à

80%).

Nous avons réalisé une série d'expérimentations avec une charge de travail qui commence à 30% et augmente uniformément jusqu'à 100% de la capacité totale du système. La capacité totale du système est définie par la somme des capacités de tous les fournisseurs.

La figure 2(a) montre la moyenne de la *satisfaction* des fournisseurs pour les trois méthodes. Nous observons que les fournisseurs sont plus satisfaits avec $Libra$ qu'avec les deux autres méthodes. De même, nous notons que la *satisfaction* des fournisseurs décroît quand la charge de travail augmente. Cela est dû au fait que les fournisseurs prennent plus en compte leur *utilisation* quand ils deviennent chargés. $Capacity\ based$ et *Mariposa-like* ne satisfont pas les fournisseurs dès le début simplement parce qu'ils allouent les tâches en considérant d'autres critères qui correspondent moins directement aux *intentions* des fournisseurs.

La figure 2(b) nous montre aussi que $Libra$ est la méthode qui satisfait le plus les clients. Nous pouvons voir dans ces résultats que la *satisfaction* des clients décroît, quand la charge de travail est élevée. Puisque les fournisseurs deviennent moins satisfaits lorsque leur charge de travail est trop élevée, $Libra$ fait plus attention à leur *satisfaction*, tout en tenant compte des clients.

En ce qui concerne l'*utilisation* des fournisseurs, comme prévu, $Capacity\ based$ assure un meilleur équilibrage des tâches parmi les fournisseurs que $Libra$ et *Mariposa-like* (c.f. Figure 2(c)). Néanmoins, nous voyons que *Mariposa-like* a des problèmes pour garantir un bon équilibre des tâches tandis $Libra$ a des performances proches à celles de $Capacity\ based$. On peut expliquer cela par le fait que les fournisseurs *très-capables* et *très-intéressants* monopolisent les tâches dans *Mariposa-like*, créant ainsi des famines chez les autres fournisseurs. Par ailleurs, nous avons pu voir pendant nos expérimentations que $Libra$ a quelques difficultés à répartir la charge entre les fournisseurs, en particulier lorsque la charge totale du système est inférieure à 40%. En revanche, quand la charge de travail augmente $Libra$ devient plus efficace car les fournisseurs commencent à s'intéresser à leur charge.

La figure 3(a) montre le nombre de départs chez les fournisseurs avec les trois méthodes. Nous voyons que $Capacity\ based$ et *Mariposa-like* perdent presque tous les fournisseurs pour toutes les charges de travail, sauf quand elle est en dessous du 30%, tandis que $Libra$ perd uniquement 28% de fournisseurs en moyenne. La figure 3(b) illustre les départs de clients. $Capacity\ based$ et *Mariposa-like* perdent en moyenne 38% et 25% de clients respectivement. Sur cette expérimentation, $Libra$ montre encore une fois son avantage en ne perdant aucun client.

Finalement, nous évaluons l'impact des départs des participants sur le temps de réponse (figure 3(c)). Le temps de réponse est défini comme le temps écoulé entre l'émission d'une tâche et la réponse. Nous pouvons voir ici que $Libra$ assure de meilleurs temps de réponse. Nous constatons aussi que $Capacity\ based$ est meilleur que *Mariposa-like* car, comme vu précédemment, *Mariposa-like* surcharge les fournisseurs qui sont les plus capables et intéressants.

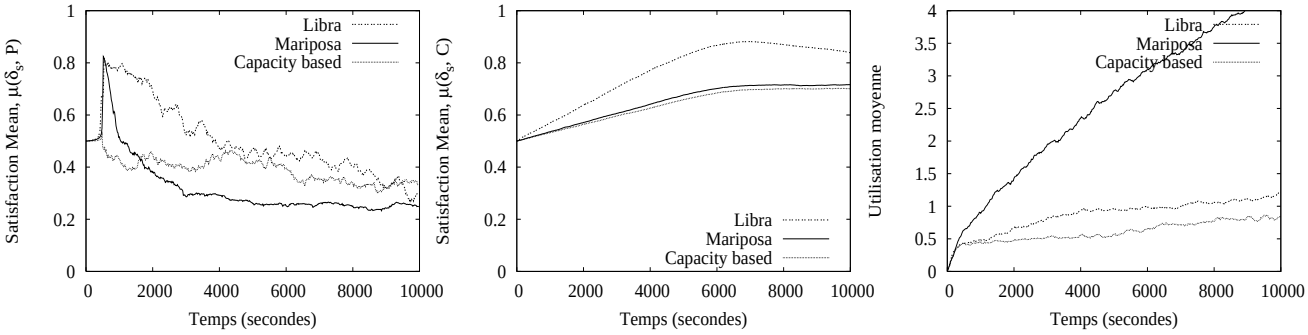
Tous ces résultats démontrent la grande adaptabilité de $Libra$ aux attentes des participants, ce qui fait que $Libra$ est fortement approprié aux environnements autonomes.

5.4 Discussion

Dans nos expérimentations, nous avons vu que les fournisseurs quittent le système par *mécontentement* dans $Capacity\ based$ et par *famine* dans *Mariposa-like*. Les fournisseurs qui décident de quitter le système avec ces deux méthodes sont pour la plupart ceux qui sont les plus capables et qui sont les plus demandés par les clients. Avec $Libra$, des fournisseurs quittent aussi le système, ceci principalement pour des raisons de *mécontentement* qui s'expliquent dans la très grande majorité des cas par des problèmes d'adéquation : ces fournisseurs sont considérés comme étant peu intéressants par les clients ou comme ayant de trop faibles capacités. D'autre part nous avons vu que les départs de clients ont aussi une certaine importance. En effet, si le nombre de tâches diminue, les fournisseurs ont moins de possibilités d'être satisfaits.

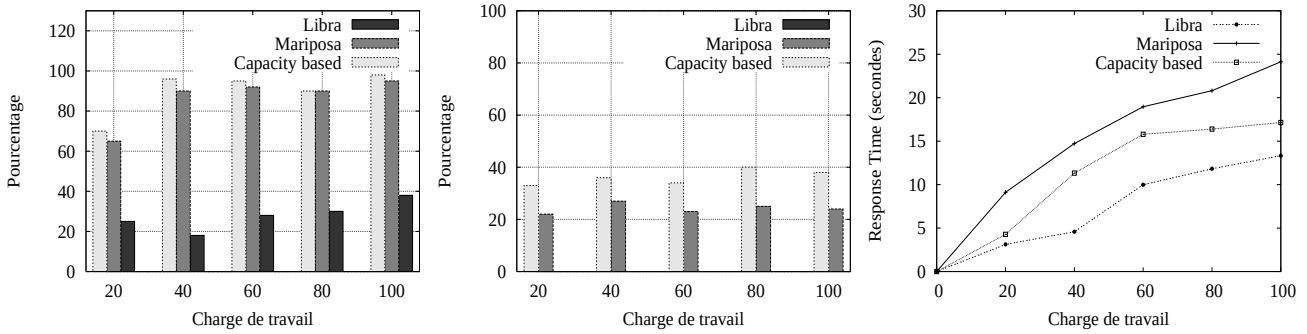
6 Conclusion

Nous avons considéré le problème de l'allocation de tâches dans des environnements ouverts où les participants ont des attentes particulières. Dans ce cadre, prendre en compte les *intentions* des participants de façon à ce que leur attentes soient satisfaites est crucial pour le bon fonctionnement d'un système. Dans cet article,



(a) Satisfaction des fournisseurs. (b) Satisfaction des clients. (c) Utilisation des Fournisseurs.

FIG. 2 – Charge croissant de 30% à 100% de la capacité du système ; agents non autorisés à quitter le système.



(a) Départs des fournisseurs. (b) Départs des clients. (c) Temps de réponse.

FIG. 3 – Charges exprimées en fonction des capacités initiales du système ; agents autorisés à quitter le système.

nous avons proposé une méthode d'allocation de tâches ($Libra$) tout en considérant et satisfaisant les *intentions* des participants.

$Libra$ diffère fortement des travaux précédents. Il arbitre entre les différents participants en se basant sur leurs *satisfactions*. Il favorise ainsi le point de vue des uns ou des autres, points de vue qui sont exprimés par les *intentions*. Cela a entre autre pour conséquence de réduire les problèmes de *famine* chez les fournisseurs.

Nous avons comparé $Libra$ avec deux méthodes importantes ($Capacity\ based$ et *Mariposa-like*) et avons montré par expérimentation que $Libra$ présente de nombreux avantages. Les résultats prouvent que $Capacity\ based$ et *Mariposa-like* perdent plus de 20% de clients alors qu'aucun client ne quitte le système avec $Libra$.

7 Travaux futurs

Dans l'approche exposée dans cet article, un médiateur quantifie la pertinence d'allouer une tâche à tel ou tel fournisseur en considérant l'*intention* du client et du fournisseur. Il favorise le moins satisfait des deux. D'autres solutions sont envisageables. Par exemple, il peut être naturel pour un médiateur de faire en sorte qu'un fournisseur (resp. un client) soit satisfait du travail de médiation. Cela nécessite l'introduction d'une notion de satisfaction par rapport à la médiation, qui est différente de la satisfaction présentée ici. Quel que soit le contexte, un fournisseur recevant n tâches aura la même satisfaction. En revanche, sa satisfaction par rapport à la médiation sera d'autant plus forte que le contexte lui sera défavorable. Autrement dit, une technique de médiation n'a aucun mérite à satisfaire un participant lorsque ses désirs sont en adéquation avec son environnement. Nous pensons intégrer cette notion dans une version future.

Lors d'expérimentations[4] faisant intervenir plusieurs médiateurs, nous avons constaté, sous

[4] Les expérimentations en question ne sont pas celles présentées dans cet article.

certaines conditions, des phénomènes d'auto-organisation : clients et fournisseurs partageant les mêmes intérêts se regroupent autour du même médiateur. Nous souhaitons explorer ce phénomène que nous n'avons pas observé avec les autres approches.

Enfin, le problème adressé dans cet article comme celui adressé par les approches économiques est de réguler un système tout en satisfaisant les participants. Nous comptons donc développer une version économique de $Libra$ pour analyser en détail les apports spécifiques de l'économie.

Remerciements

Ce travail a été en partie financé par ARA "Massive Data" du ministère français pour la recherche (projets MDP2P et Respire) et le projet européen Grid4All. Le deuxième auteur de cet article est supporté par le Conseil National de Science et Technologie du Mexique (CONACyT).

Références

[1] R. Miller, editor. *Special Issue on Integration Management. IEEE Data Eng. Bull.*, 25(3), 2002.

[2] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced Allocations. *SIAM J. Comp.*, 29(1), 1999.

[3] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems. In S. H. Clearwater, editor, *Market-Based Control : A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[4] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed computer systems. In *ICDCS Conf.*, 1988.

[5] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *Procs. of the VLDB Conf.*, 2004.

[6] P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. In *Procs. of CoopIS Conf.*, 2004.

[7] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Procs. of the WWW Conf.*, 2003.

[8] E. P. Markatos. Tracing a Large-Scale Peer to Peer System : An Hour in the Life of Gnutella. In *Procs. of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[9] R. Mirchandaney, D. F. Towsley, and J. A. Stankovic. Adaptive Load Sharing in Heterogeneous Distributed Systems. *Journal of Parallel and Distributed Computing*, 9(4), 1990.

[10] M. H. Nodine, W. Bohrer, and A. H. Ngu. Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In *Procs. of the ICDE Conf.*, 1999.

[11] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. Satisfaction Based Query Load Balancing. In *Procs. of the CoopIS Conf.*, 2006.

[12] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. KnBest - A Balanced Request Allocation Method for Distributed Information Systems. In *Procs. of the DASFAA Conf.*, 2007.

[13] E. Rahm and R. Marek. Dynamic Multi-Resource Load Balancing in Parallel Database Systems. In *Procs. of the VLDB Conf.*, 1995.

[14] M. Roth and P. Schwarz. Don't Scrap It ! Wrap It ! A Wrapper Architecture for Legacy Data Sources. In *Procs. of the VLDB Conf.*, 1997.

[15] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Procs. of the Multimedia Computing and Networking Conf.*, 2002.

[16] N. G. Shivaratri, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems. *IEEE Computer Journal*, 25(12), 1992.

[17] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa : A Wide-Area Distributed Database System. *VLDB Journal*, 5(1), 1996.